# Sinclair ZX Spectrum Userclub

Clubinfo Nr. 2 (85) 01. Februar 1991

Rolf Knorre - Postfach 20 01 02 - 5600 Wuppertal 2
Telefon 0202 / 55 66 89
Stadtpsarkasse Wuppertal (BLZ 33050000) Konto Nr. 853382

Hallo Freunde,

als Ausgleich für das verspätete Januar - Info kommt die Februar - Ausgabe diesmal einige Tage früher bei Euch an. Im nächsten Monat läuft dann wieder alles normal weiter.

Da es sonst nichts zu berichten gibt, wünsche ich allen Mitgliedern viel Spaß mit dem mittlerweile 85. Clubinfo.

Rolf Knorre

## Text-o-Mat (13)

Random-Access- Datei für OPUS-Discovery.

Diese brandneue Erweiterung hat mich schon seit Jahren beschäftigt und mich wie die Katze um den heißen Brei schleichen lassen. Zwar hatte mir damals Michael Töpfer eine Mc- Routine verehrt, die es gestattet, aus Tasword II- Files Randomfiles zu machen, doch war nicht ersichtlich, wie diese Files wieder gelesen werden könnten.
Als ich jetzt eine Erweiterung für "LAST WORD" vom SDC einsehen konnte, in der etwas aufwendig eine Random- Datei eingebaut ist, und ich diese herausoperieren und extra verwenden konnte, fiel mir wieder diese Routine von Töpfer ein, die bisher ungenutzt herumlag. Die Schwierigkeit, die mir lange nicht lösbar schien, war, wie an die einzelnen Datensätze heranzukommen sei, wenn ihre Nummer nicht bekannt ist. Dies ist bei LAST WORD irgendwie gelöst und funktioniert auch. Aber soviel Platz habe ich in meinem Text-o-Mat nicht. Jenes Programm arbeitet 7 Variablen zu je 32 Bwstaben, meine Routine hat zunächst einen Satz mit 64 Buch-

Die Lösung war nun, das Randomfile ausdrucken zu lassen und dabei jedem Datensatz die richtige Nummer zuzuordnen. Dann kann durch einfache Eingabe der Nummer, der Datensatz aufgerufen werden. Das Verfahren ist zwar nicht sehr witzig für Telefonnummern denn die könnte man dann ja gleich auf die Liste drucken, aber für Adressen oder Pferdedaten ist das schon etwas!
Damit ist der Datensatz also als Basicvariable im Text-o-Mat.
Wie kommt er aber (als mc-code) in den Editor? So, wie ich die Adresse für einen Brief im Text-o-Mat eingebe (INFO 11/90) und zunächst mittels Routine 55149 in Zeile 1 auf Helpseite 1 ablege dann die einzelnen Daten an die richtige Stelle am Briefkopf mittels Routine 55308 sende, genauso mache ich das jetzt mit dem Random- Datensatz.
Wenn ich eine Adresse schreibe: Brightmode- a (Zeile 8234),werde ich gefragt, ob ich ein Mitglied anschreiben will, gebe da die Nummer ein, die ich aus einer automatisch gedruckten Liste entnehme, und die ganze Adresse steht schon richtig da.
Wenn ich die Randomdaten aber in einer beliebigen Textzeile verwenden will, ist der Einstieg wieder mal EXT-r- AT (Z.3025/29). Der Computer fragt hier, welche Datei er verwenden soll und geht dann selbst- ständig zum Befehl EXT-r- < (S.INFO 6/90) . So sind 2 Möglichkeiten der Verwendung integriert, einmal als Adresse, dann zum Einsetzen an beliebiger Stelle.
Für die Einrichtung eines Random- Access- Files gibt es nun 2 Möglichkeiten. 1) Vom Text-o-Mat aus mittels Bright- m (Z.8098) abspeichern. Die Routine von Töpfer sitzt bei 31970 vor dem Tasword-mc und hat nur 30 bytes (sicher ein Meisterwerk. Ich liebe so klei- ne Routinen!) Man kann sie zuladen (sogar nachträglich), könnte sie aber auch ins Tasword-mc einbinden. 2) Man kann das Tasword- File in ein Hilfsprogramm einspielen und dort konvertieren. Dieses Programm kann auch lesen und die Listen mit Vom Text-o-Mat aus kann man nur 320 Datensätze direkt übertragen man kann aber 523 Sätze zusammenleimen und dann im Hilfsprogramm konvertieren. Für größere Sätze müßte die MC- Routine tiefer gesetzt werden (POKE Anfang +9/10, Beginn der Daten, ca. 630 Sätze bzw. weitere Sätze auf andere Art eingebracht werden. Dann könnte ein File den Speicherplatz einer ganzen Diskette einnehmen. Welch' eine bedeutsame Erweiterung für den SPECTRUM!
Korrektur und Eingabe neuer Datensätze ist hier zunächst nicht nötig, denn diese kann ja jeweils im Text-o-Mat erfolgen, worauf das ganze File dann neu konvertiert wird. Im Hilfsprogramm kann man das aber vorsehen.
Natürlich kann man auch eine umfangreichere Randomdatei mit dem Text-o-Mat in ähnlicher Weise verbinden, z.B.die des "LAST WORD" mit 7 Zeilen. Wenn anderes weggelassen wird, ist auch das prinzipiell möglich. Für mich reicht aber zunächst diese Version mit 64 Buchstaben pro Datensatz. Mit POKE 31983, kann man die Routine aber bis 256 Zeichen erweitern (OPEN#4 Befehle anpassen,257).

Herbert Hartig
Postfach 147,8938 Buchloe

```
   1 GO TO PI
   2 CLEAR 31969: LOAD *1;"tw2/rnd_mc"CODE 31970,30
   5 CLS : PRINT "RANDOM-ACCESS-DATEI"'''"Hilfsprogramm fuer Verw
endung   mit Text-o-Mat:"''''"von Tasword-File nach Datei....k"''
"Lesen der Datei.............1"''''"Druck der Datei mit Nummer.
....d": PAUSE NOT PI: CLS : GO TO 5+5*(INKEY$="k")+25*(INKEY$="1
")+47*(INKEY$="d")
  12 CLS #: PRINT AT 0,0;" Tasword 2 -> Random Konverter  ";" 2/
1989 by Axel Toepfer": DIM t$(10): INPUT "Name des TW2 Textes: "
;t$: IF t$(1)="" THEN LET t$="§"
  13 IF t$(1)="9" THEN CAT 1: PAUSE NOT PI: GO TO 12
  14 LOAD *1;t$CODE 32e3
  15 DIM d$(10): INPUT "Name der Datei: "; LINE d$
  16 PRINT AT 20,0;"bitte Diskette drin lassen": OPEN #4;1;t$IN
: RANDOMIZE CODE INKEY$#4: LET l=CODE INKEY$#4+256*CODE INKEY$#4
: CLOSE #4
  18 LET z=(1/64): POKE 31977,INT (z/256): POKE 31976,z-(256*PEE
K 31977)
  19 PRINT "Diskette wechseln?": PAUSE 0
  20 OPEN #4;1;d$RND65,z: RANDOMIZE USR 31970: CLOSE #4
  22 PRINT AT 20,0;"Noch einen Text konvertieren ?"
  24 IF INKEY$="" THEN GO TO 24
  26 IF INKEY$="N" OR INKEY$="n" THEN CLS #: GO TO 5
  28 RUN 12
  35 INPUT "Dateiname: ";n$
  37 OPEN #4;1;n$RND65
  39 INPUT "welcher Satz";n: PRINT n;
  40POINT #4;n
  42 INPUT #4;f$
  44 PRINT f$
  46 INPUT "noch ein Satz?";r$
  48 IF r$="j" THEN GO TO 39
  50 CLOSE #4: GO TO 5
  52 CLEAR #: OPEN #3;"t"
  54 INPUT "Dateiname: ";n$;"Anzahl Datens?";d
  56 OPEN #4;1;n$RND65
  58 FOR n=1 TO d: LPRINT n;" ( ";
  60POINT #4;n
  62 INPUT #4;f$: LPRINT f$
  64 NEXT n
  66 CLOSE #4
  68 GO TO 5
9900 SAVE *1;"run" LINE 2


3025 IF INKEY$=" AND " THEN INPUT "Name Dat.";j$: GO SUB VAL "60
13": LET t$="("
3029 IF INKEY$="(" OR t$="(" THEN GO SUB VAL "7010": LET b=VAL "
56128"+VAL "64"*ze: GO SUB VAL "7015": PRINT USR VAL "55308",b,V
AL "32e3"+z1,VAL "64"*zu: GO TO VAL "3112"
8079 PRINT USR VAL "55308",VAL "56129"+VAL "64"*ze,VAL "32640",V
AL "23": PRINT USR VAL "55308",VAL "56152"+VAL "64"*ze,VAL "3270
4",VAL "20": PRINT USR VAL "55308",VAL "56172"+VAL "64"*ze,VAL "
32832";VAL "20": RETURN
8098 DIM d$(VAL "10"): INPUT "Dateiname:";d$: LET z=(a/VAL "64")
: POKE VAL "31977",INT (z/VAL "256"): POKE VAL "31976",z-(VAL "2
56"*PEEK VAL "31977")
8099 OPEN #VAL "4";SQR PI;d$RNDVAL "65",z+SGN PI: RANDOMIZE USR
VAL "31970": CLOSE #VAL "4": GO TO PI
8234 GO SUB VAL "6010": INPUT "An:";e$;"Vorn.";f$;"Name";g$;"Str
.";h$;"Ort";i$;"Tit.?(j)";j$;"Kopf?";k$: GO SUB VAL "7110"
```



52.4.205,1.22.1.15.0,33.0.125,197.6.64.126,215.35.16.251,193.62.

# THE SECRETS OF BETA DOS CALLS

by: Hendrik BROOTHAERS
AZALEALAAN 13
B-2630 AARTSELAAR
BELGIUM
Tel: intnl.code-32-3-887-83-23

ins Deutsche übersetzt und erweitert: Bernhard LUTZ
bei Wünschel
Obermühlstr. 24
6729 Bellheim
Tel. 07272/6868

-Es folgt eine Erklärung nützlicher Subroutinen des BETA DOS
-Die Routinen können mit Machinensprache angesprochen werden
-Sie ermöglichen es auf einfache Weise das DOS zu benutzen, was durch Beispiele demonstriert wird.
-Die Routinen sind nur in den DOS Versionen 4.xx und 5.xx vorhanden, nicht in den DOS Versionen 3.xx
-Als erstes folgt eine Erklärung wie die Routinen aufgerufen werden (mittels CALL), und was sie bewirken.
-All diese Routinen werden über eine zentrale Adresse angesprungen. Die CALL Adresse ist #3BFD (15357). Wenn man diese Adressse aufruft muß die Nummer der gewünschten Routine im C Register stehen. Die Inhalte der anderen Register sowie einiger DOS-Variablen werden zusätzlich als Parameter und zur Steuerung der Routinen benutzt.
-FÜR DOS VERSION 5.XX GILT EINE ANDERE ZENTRALE ADRESSE UND ANDERE ADRESSEN UM DAS ROM EIN- BZW. AUSZUSCHALTEN.

ALLE FOLGENDEN BEISPIELE GELTEN NUR FüR DOS VERSION 4.XX !!

Um die Routinen ansprechen zu können muß als erstes das DOS eingeschaltet werden, dies wird durch einen CALL #3C06 (15366) gefolgt durch ein PUSH HL erreicht. (Das PUSH HL legt eine RETURN Adresse auf den Stack, die automatisch unser DOS wieder abschaltet und zu der dann automatisch gesprungen wird wenn wir unseren Code mittels eines RETurn verlassen.
Als nächstes müssen wir nun die benötigten Register laden (wenn das nicht schon getan wurde), dann das C Regsiter mit der benötigten Routinen Nummer laden und abschliessend einen CALL #3BFD gefolgt von einem RETurn ausführen.

## Erklärung einiger von den Routinen benutzter DOS Variablen

#5CD1(low)/#5CD2(high) = Auto RUN Zeilen Nummer.
(23761/2) Wird benutzt wenn ein BASIC File mittels Routine 12 auf Disk gespeichert wird.

#5CD2 (23762) Muß den Code des Array (Feld-) Zeichens enthalten wenn ein DATA File (Buchstaben oder Zahlenfeld) mittels Routine 14 geladen wird.

#5CD7/8 (23767/8) Enthält die Buffer-(Lade-) Adresse für das Lesen oder Schreiben eines Sektors. Wird von Routine 14 benutzt, wenn (#5D10) (23824) ungleich 0 ist.

#5CDD bis #5CE4 (23773 - 23780) Filename (8 Zeichen lang)

#5CE5 (23781) Filetype (B) Basic (C) Code (D) Data

#5CE6/7 (23782) Gesamtlänge für BASIC
(23782/3) Ladeadresse fär CODE oder DATA

#5CE8/9 Programm-Länge von BASIC Files
(23784/5) Byte-Länge für CODE oder DATA Files

#5CEA (23786) Filelänge in Sektoren

#5CEB/C Sektor und Track wo das File auf der Diskette
(23787/8) beginnt

#5D0F (23823) Nummer eines Files im Katalog (wird von Routine 10 benutzt)

#5D10 (23824) Kontrollbyte für Routine 14 (siehe Text)

**********

## R O U T I N E N B E S C H R E I B U N G

Die Routinen sind numeriert von 0 bis 20 (oder in HEX #00 - #14)

### ROUTINE 0 (# 00)

Das gewählte Laufwerk wird auf Track 0 zurückgestellt und die BREAK-Tasten abgefragt.

### ROUTINE 1 (# 01)

Das Laufwerk angewählt, dessen Nummer im A Register ist.

### ROUTINE 2 (# 02)

Positioniert den Kopf zu der Track Nummer, die A Register steht.

### ROUTINE 3 (# 03)

Speichert A-Register in #5CFF ( = Sektor zum Lesen / Schreiben)

### ROUTINE 4 (# 04)

Speichert HL Register in #5D00 ( = Bufferadresse)
Beachte: Routine 3 und 4 haben keine spezielle Funktion an sich, sie werden aber innerhalb anderer Routinen aufgerufen.

### ROUTINE 5 (# 05)

LESEN von Diskette. Jeder Sektor/Track und jede Anzahl von Blocks (entsprechend Sektoren)
Benutzte Register: B = Anzahl der zu lesenden Sektoren
DE = Track/Sektor fär zu lesende Daten
HL = Bufferadresse für zu lesende Daten

### ROUTINE 6 (# 06)

SCHREIBEN auf Diskette. Jeder Sektor/Track und jede Anzahl von Sektoren.
Benuztet Register: B = Anzahl der zu schreibenden Sektoren
DE = Track/Sektor für zu schreibende Daten
HL = Bufferadresse für zu schreibende Daten

ROUTINE 7 (# 07)
==========
CATalog zum Strom (Stream) dessen Nummer im A Register steht.

ROUTINE 8 (# 0 8)
==========
LIEST File Info von Track Zero in die DOS Variablen #5CDD <--> #5CEC. Für das File dessen Nummer im A Regsiter steht. (16 Bytes werden übertragen).

ROUTINE 9 (# 09)
==========
SCHREIBT File Info aus den DOS Variablen #5CDD <--> #5CEC auf Diskette. Für das File dessen Nummer im A Register steht. (16 Bytes werden übertragen).

ROUTINE 10 (# 0A)
===========
SUCHT Disk-Katalog nach dem File ab, dessen Name und Typ in den DOS Variablen #5CDD bis #5CE5. Nach Beendigung enthält Adresse #5D0F die Nummer des Files innerhalb des Katalogs bzw. #FF when das File nicht gefunden wurde.

ROUTINE 11 (# 0B)
===========
SAVEd ein Nicht-Basic File. Filename und Typ müssen in den DOS Variablen #5CDD bis #5CE5 stehen.
DE = Länge in Bytes HL= Startadresse

ROUTINE 12 (# 0C)
===========
SAVEn eines BASIC File. Filename und Typ müssen in den DOS Variablen #5CDD bis #5CE5 stehen. In #5CD1(low)/#5CD2(high) muß die Auto RUN Zeilennummer stehen.

ROUTINE 13-15-16-17 (# 0D-0F-10-11)
====================
Diese Routinen sind ein und dasselbe, sie werden innerhalb des DOS als Ausgang (EXIT-Routine) für die anderen Routinen benutzt.

ROUTINE 14 (# 0E) (wird später ausführlich erklährt)
==========
LOAD (BASIC-CODE-DATA). Die Diskettenoperation hängt von den Inhalten der Speicherzellen #5CD2 ,#5CD7 ,#5D10 und den Registern A - HL - DE ab. Filename und Typ müssen in den DOS Variablen #5CDD bis #5CE5 stehen.

ROUTINE 18 (# 12)
==========
ERASE des Files, dessen Name und Typ in den DOS Variablen #5CDD bis #5CE5 steht.

ROUTINE 19 (# 13)
==========
LDIR Speicher in die DOS Variablen #5CDD bis #5CEC. HL ist der Speicherzeiger.

ROUTINE 20 (# 14)
==========
LDIR der
DOS Variablen #5CDD bis #5CEC in den Speicher. HL ist der Speicherzeiger.

**********

Details von Routine 14 (# 0E)
==========

Routine 14 ist die am meisten komplexe Routine, sie wird benutzt um Files zu laden. Ich werde nun die verschiedenen Fälle erklären. Man muß IMMER sicherstellen, das der Filename und Typ in den DOS Variablen #5CDD bis #5CE5 steht, bevor die Routine aufgerufen wird, sonst wird die Meldung "NO FILE" ausgegeben und die Operation abgebrochen.

Für ein BASIC File:

#5D10=#00 A Register =#00 dann CALL Routine 14
-es muß genug Platz unter RAMTOP frei sein
-jedes vorhandene BASIC wird gelöscht

Fär ein DATA File (Zahlen oder Zeichen-Feld)

#5D10=#00 A Register =#00 HL Register =#0000
#5CD2 = Feld-Name z.B. #81 für Zahlen-Array "a"
#C1 für Zeichen-Array "a$"
(Siehe SPECTRUM handbook pages 167 / 168)
Die Routine DIMensioniert das Array (Feld) bevor es die Daten einlädt.

Fär ein CODE File

#5D10 hat folgende Kontrolle über Routine 14 :
#5D10 =#00 = LADEN eines kompletten Files
#5D10 =#FF = LADEN eines einzelnen Sektors
#5D10 verschieden von #00 und #FF = SCHREIBT einen Sektor auf Disk

-mit #5D10 =#00

-A Register =#00 LÄDT ein File zu der Adresse, von der es abgesaved wurde.
-A Register =#03 LÄDT die ersten Sektoren eines Files zu der Adresse die im HL Register steht.
D Register = Anzahl der zuladenden Sektoren
-A Reg. weder #00 noch #03 = LOAD eines Files an die Adresse die im HL Register steht.

-mit #5D10 =#FF

LOAD eines Sektors dessen Nummer im L Register steht zu der Adresse die in den DOS Variablen #5CD7/8 steht.

Beispiel: wenn L = 5 wird der fünfte Sektor geladen

-mit #5D10 verschieden von #00 und von #FF

SCHREIBT einen Sektor in ein File. Wie oben muß das L Register mit der Sektorennummer sowie die Speicheradresse in den DOS Variablen #5CD7/8 stehen bevor Routine 14 aufgerufen wird.

**********

ROUTINES Ext. (by M.J. SMITH) ACHTUNG, die folgenden vier Routinen sind nur im TRDOS 5.xx vorhanden !!!

ROUTINE 21 (# 15)

VERIFY des gerade gewählten Tracks (von 0 bis 79). Vor dem Anspringen der Routine B-Register = 1 setzen (=Anzahl der Tracks), A Reg.=Track, D=Track, E=Sektor CALL #3D13.
Zurück erhält man im BC Register den Error-Code. (Wenn Error-Code =0 dann OK.)

ROUTINE 22 (#16)
==========

Wählt Seite 0 eines Double sided Disk Systems an (Grundeinstellung für Single sided Laufwerke). Aufruf: C=#16 dann CALL #3D13 (=ZENTRALE ADRESSE FÜR DOS VERSION 5.XX)

ROUTINE 23 (#17)
==========

Wählt Seite 1 für Double sided System. C=#17 dann CALL #3D13

ROUTINE 24 (#18)

Diese Routine überprüft das Directory des gerade gewählten Laufwerks und gibt den Drive Status mittels der Error-Codes 0-12 aus. Alternativ steht der Error-Code auch im BC-Register.

==========

DOS ROUTINES Examples (Beispiele - by H.Broothaers) (DOS 4.XX)

1. Folgendes Beispiel sendet einen Katalog auf den Bildschirm:

BEISPIEL EINS

```
CALL #3C06      ;DOS einschalten
PUSH HL         ;DOS-AUS RETurn Adresse auf den Stack legen
LD A,#02        ;wählt Strom (Stream) 2 (Screen)
LD C,#07        ;selektiert Routine 7 (Katalog)
CALL #3BFD      ;CALL DOS um Routine 7 auszuführen
RET             ;Return via DOS-Aus Aufruf
```

2. Die nächsten Beispiele sind eine Kombination von drei Sachen:

Bevor der Code aufgerufen wird muß das DE Register auf einen Speicherbereich zeigen, wo Filename und Typ abgelegt wurden.
Als erstes wird der Filename in die DOS Variablen geschrieben (mit Routine 19), dann wird der Diskkatalog nach dem entsprechenden File abgesucht (mit Routine 10).
Wenn das File nicht existiert enthält #5D0F den Wert #FF, wenn es existiert enthält #5D0F die Nummer des Files innerhalb des Katalogs.

Zum Schluß wird dann noch die Informationen (Filelänge, Sektor, Track über das File von Track Zero in die DOS Variablen gebracht (mittels Routine 8).

BEISPIEL ZWEI

```
CALL #3C06      ;DOS einschalten
PUSH HL         ;DOS-AUS (OFF) RETurn Adresse auf den Stack legen
EX DE,HL        ;HL zeigt nun auf den Filenamen
LD C,#13        ;wählt Routine 19 (#13)
CALL #3BFD      ;CALL Rout. 19, Filename in die DOS Variablen
LD A,#43        ;Charakter (Zeichen) "C" 67 (#43)
LD (#5CE5),A    ;setzt Filetyp in den DOS Variablen
LD C,#0A        ;wählt DOS Routine 19 (#0A)
CALL #3BFD      ;CALL Rout. 10, sucht Katalog nach File ab
LD A,(#5D0F)    ;hole das Resultat ins A Register
CP #FF          ;prüfe ob File existiert
RET Z           ;RETurn wenn keines existiert
LD C,#08        ;wähle DOS Rout.8
CALL #3BFD      ;CALL Rout.8 File Info in DOS Variablen ablegen
RET
```

3. Dieses Beispiel zeigt wie die Disk-routinen aus andren Routinen heraus aufgerufen werden können.
Die Zeilen 1 bis 10 (eigenlich Labels) zeigen eine Routine um ein File von Disk zu laden. Wenn diese Routine bei 1 angesprungen wird, wird das File in den Speicher ab #A000 geladen. Wenn die Routine bei 2 angesprungen wird, läft das Programm das gewünschte File ab der Adresse, auf die das DE Registerpaar zeigt.

BEISPIEL DREI

```
 1 LD DE,#A000    ;DE mit der Bufferadresse 40960 (#4000) laden
 2 CALL #3C06     ;DOS einschalten
 3 PUSH HL        ;RETurnadresse speichern
 4 CALL 32        ;CALL Rout. um  Drive zu wählen (32=Zeile 32!)
 5 EX DE,HL       ;Bufferadresse ins HL Registerpaar
 6 XOR A          ;A Register gleich 0 setzen
 7 LD (#5D10),A   ;(#5D10) mit 0 laden (siehe Rout.14 Beschreib.)
 8 LD A,1         ;A Register gleich 1 (Lade ab Adresse in HL)
 9 LD C,14        ;wähle Rout. 14 um zu laden
10 JR 24          ;führe Rt. 14 aus und kehre über DOS-OFF zurück
```

Die Zeilen (hier Labels !!) 20 bis 25 ergeben einen Katalog auf dem Bildschirm (Screen)

```
20 CALL #3C06      ;schalte DOS ein
21 PUSH HL         ;sichere die RETurn Adresse auf dem Stack
22 LD A,#02        ;wähle Stream 2= Screen
23 LD C,#07        ;wähle Routine 7= Katalog
24 CALL #3BFD      ;führe die Rout. mit Kennummer im C-Reg. aus
25 RET             ;RETurn
```

Die Zeilen 30 bis 42 wählen das Drive, dessen Nummer in der Speicherzelle #A123 steht, an und rufen die Katalog Routine auf.

```
30 CALL #3C06      ;schalte DOS an
31 PUSH HL         ;lege die RET-Adr. (via DOS-OFF) auf den Stack
32 PUSH DE         ;sichere Wert
33 PUSH BC         ;sichere Wert
34 PUSH AF         ;sichere Wert
35 LD A,(#A123)    ;lade A Register mit der Drive-Nummer
36 LD C,#01        ;wähle DOS Routine 1= Laufwerk selektieren
37 CALL 24         ;wähle Drive an (via CALL zu Zeile 24)
38 CALL 22         ;mache einen Katalog
39 POP AF          ;Wert zurückholen
40 POP BC          ;Wert zurückholen
41 POP DE          ;Wert zurüvkholen
42 RET             ;RETurn
```

DIE GEHEIMNISSE DES TRACK 0 (=ZERO) des Beta-Disk Systems

(von Hendrik Broothaers, ins Deutsche übersetzt von:
Bernhard Lutz,Hammerstrasse 35,6729 Bellheim,Tel. 07272/71444)

Das folgende gilt für ein "double density"-Laufwerk:

Als erstes einige grundsätzliche Dinge:
-Der Aufbau von Track Zero ist derselbe für jedes Format, 40 od. 80 Track, Single oder double sided. Das ist natürlich wichtig, da wir ja zuerst das Format wissen müssen, bevor wir uns die Informationen zugänglich machen können.
-Jeder Track (d.h. jede Spur) ist in 16 Sektoren unterteilt.
-Jeder Sector ist 256 Bytes groß.

Benutzen des Track Zero:
-Es sind nur 9 der 16 Sektoren belegt.
-Sector 8 (wir beginnen bei 0 zu zählen, also eigentlich der neunte Sector), enthält alle Informationen der Disk wie Diskname, Passwort (nur bei älteren DOS-Versionen), Anzahl der Files usw.
-Die Sektoren 0 bis 7 enthalten die Informationen über die Files auf der Diskette.
-Jedes File belegt dort 16 Bytes, sodaß wir die Informationen von 16 Files innerhalb eines Sektors ablegen können.
-Weil wir nun die ersten 8 Sektoren (0-7) benutzen, können wir ein Maximum von 128 Files in Track Zero unterbringen.

Sektor 8 enthält die Grunddaten der Diskette:
-Dabei werden nur die letzten 31 Bytes des Sektors benutzt, die ersten 225 Bytes sind alle 0.

| BYTE | Bedeutung |
|---|---|
| 225 | sektor } Erster freier Sektor und |
| 226 | track } erster freier Track auf der Disk |
| 227 | format typ (siehe nachfolgende Tabelle) |
| 228 | Anzahl der Files (inclusive gelöschter Files) |
| 229 | } Anzahl der freien Sektoren |
| 230 | } auf der Diskette |
| 231 | 16 |
| 232 | 0 |
| 233 | 0 |
| 234 | bis 242 Passwort (9 Zeichen lang) |
| 243 | 0 |
| 244 | Anzahl der gelöschten Files |
| 245 | bis 253 Name der Diskette (9 Zeichen lang) |
| 254 | 0 |
| 255 | 0 |

Tabelle für Byte 227:

| | | |
|---|---|---|
| 22 | = 80 track/double side | = 2560 Sektoren |
| 23 | = 40 track/double side | = 1280 Sektoren |
| 24 | = 80 track/single side | = 1280 Sektoren |
| 25 | = 40 track/single side | = 640 Sektoren |

Sektor 0 bis 7 enthalten die File Informationen
-Wie schon erwähnt, benutzt jedes File 16 Bytes
-Ein paar dieser Bytes haben verschiedene Bedeutung, abhängig vom Typ des Files (Basic, Code oder Data)

BYTE-Bedeutung:

0 bis 7 = File-Name (8 Zeichen lang)

8 = File-Typ (# (CHR$ 35) = Data-File seq. od. random
B (CHR$ 66) = Basic Programm
C (CHR$ 67) = Code-File
D (CHR$ 68) = Data-File (Array))

9 und 10 = LOAD-Adresse eines Code-Files
= Gesamtlänge eines Basic-Files (incl. Variablen)
= Adresse eines Data-Files

11 und 12 = Autostart-Adresse eines Code-Files
= Programmlänge eines Basic-Programms
= Array-Länge für ein Data-File

13 = Länge in Sektoren

14 sector }
} wo das File auf der Diskette steht.
15 track }

Bemerkungen:
-Bei einem gelöschten File ist das erste Byte (0) des Namens gleich 1.
-Die RUN-Zeilennummer eines Basic-Programmes ist direkt hinter dem Basic-Programm auf der Diskette, und NICHT in Track Zero.
-Das Ende des Directory wird dadurch angezeigt, daß das erste Byte (0) eines Programmnamens gleich 0 ist.

Orig. von H.Broothaers, uebersetzt von B.Lutz,Tel.07272-6868

Im Nachfolgenden werden die direkten Einsprungadressen fuer die verschiedenen DOS Versionen angegeben. Ich moechte euch allerdings bitten, wenn ihr Programme schreibt, IMMER die in einer der vorhergehenden Beta-Disk Folgen beschriebenen Zentral-adressen zu benutzen, damit das ganze auch jeweils bei anderen Usern mit einer von euch verschiedenen DOS Version richtig laeuft. Vielen Dank.

Direkte Call Adressen fuer die versch. ROM Versionen

| Function | 3.0 | 4.03<br>4.04 | 4.06 | 4.07<br>4.09 | 4.10 | 4.1 | 4.11<br>4.12 |
|---|---|---|---|---|---|---|---|
| DOS ROM EIN | 3CAAh | 15467 | 15467 | 15467 | 15467 | 15467 | 15467 |
| DISK LESEN | 3C09h | 11958 | 11992 | 11958 | 11991 | 11934 | 11990 |
| DISK SCHREIB. | 3C0Ch | 11974 | 12008 | 11974 | 12007 | 11950 | 12027 |
| DOS ROM AUS | 3CBEh | 15484 | 15484 | 15484 | 15484 | 15484 | 15484 |

Vor dem Lese- bzw. Schreib-CALL die Register wie folgt laden:

DE = D =Track, E = Sector
HL = Buffer addresse
BC = B=Anzahl der Sectoren, C=0

ACHTUNG: Nicht vergessen das ROM vorher ein und spaeter wieder auszuschalten !!!

Manche werden das Ganze fuer ein bischen langsam halten (was ich nicht verstehe, Haha), also sollte folgende Routine benutzt werden um SCHNELLES Schreiben und Lesen zu erreichen:

Dieses Beispiel ist fuer Version 3.0

```
CALL 3CAAh
ld   HL,Bufferaddresse
ld   D,Start track
ld   E,Start sector
ld   a,Anzahl der Sectoren (was unterschiedlich zu oben ist).
CALL LOADER ROUTINE
hier kann es dann mit dem eigenen Programm weitergehen...
```

LOADER ROUTINE:

```
LD   B,1  (Immer nur ein Sector)
ld   C,0  (C is immer 0 (wie oben))_____________________
PUSH AF
PUSH BC
PUSH DE
PUSH HL            (stand nicht im Originalbericht)
Call 3C09h (Muss 3C0Ch wenn man Schreiben will !!)
POP  HL            (stand nicht im Originalbericht)
INC  H             (stand nicht im Originalbericht)
POP  DE
POP  BC
```

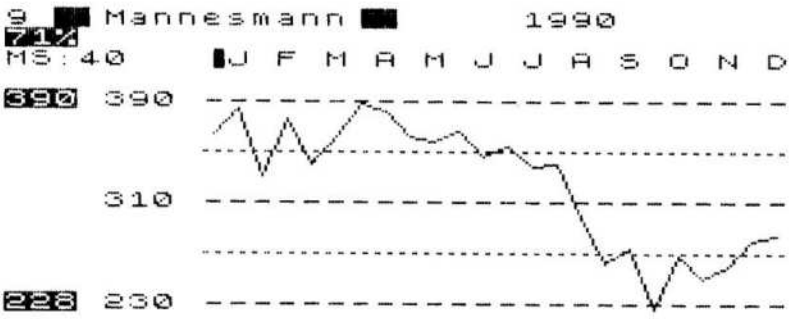
# Spectrum User Club Wuppertal

Sorry!
An dieser Seite war nichts mehr zu restaurieren,
weil sie nur aus einem verschobenen
Wuppertal-Logo bestanden hat.
Daher gild die Seite SUC-Wup85-14 als verschollen

Ludwig Halbritter
Augsburger Str. 4
8935 Fischach
Tel.: 08236/495

Fischach, 29.12.90

Hallo Spectrum-User,
mit Aktien-Trend möchte ich ein eigenes Programm vorstellen. Wer an der Börse spekuliert braucht Informationen über die Kursentwicklung der Aktien. Das Programm Aktien-Trend zeigt fortlaufend die Kursenentwicklung einer Aktie im Zeitraum von 12 Monaten in grafischer Form. Dazu ein Beispiel:




Die Kopfzeile zeigt den Namen der Aktie und das Jahr. Darunter wird der Kursunterschied in Prozent angezeigt. MS ist der Maßstab der Grafik von Linie zu Linie. Der Maßstab wird je nach Differenz von Höchst- und Niedrigstkurs (Invers ganz links angezeigt) automatisch errechnet. In der Zeile über der Grafik sind die Anfangsbuchstaben der Monate. Je Monat wird zweimal der Kurs eingegeben, das Jahr ist also in 24 Halbmonate unterteilt. Im oben gezeigten Beispiel erfolgte die letzte Kurseingabe für die erste Dezemberhälfte (Grafik ganz rechts). Bei der nächsten Eingabe wandert die Grafik nach links und die zweite Dezemberhälfte 1990 schließt sich rechts an. Es können 160 Aktien gespeichert werden. Eine Suchroutine ermöglicht die Auswahl von Aktien, bei denen der letzte Kurs eine bestimmte Prozentspanne unterhalb des Höchstkurses liegt.
Interessenten können das Programm von mir erhalten.

Zu verkaufen: Spectrum 128K (defekt, stürzt ca. 2-3 Minuten nach dem Einschalten ab) DM 50,--

Suche Strategie- und Simulationsspiele, besonders 10th FRAME und TRACK SUIT MANAGER. Bezahle Spitzenpreise!

Mit freundlichen Grüßen

L. Halbritter

Hubert Kracher
Kirchdorfer Str. 5
8201 Großholzhausen

********************
* DOUBLE 'H' SOFTWARE *
********************

Harald Lack
Heidenauer Str. 5
8201 Raubling

Liebe Specci-Freunde!

## Der Befehlssatz des Zilog Z 80 / Teil 7

### LD r,(IX + d)

Lade das Register r indirekt aus der indiziert adressierten Speicherzelle (IX + d).

Der Inhalt der Speicherzelle, die durch das Register IX plus dem gegebenen Offset adressiert wird, wird in das Register r geladen.

Beispiel: LD E,(IX + 5)

### LD r,(IY + d)

Lade das Register r indirekt aus der indiziert adressierten Speicherzelle (IY + d).

Der Inhalt der Speicherzelle, die durch das Register IY plus dem gegebenen Offset adressiert wird, wird in das Register r geladen.

Beispiel: LD A,(IY + 2)

### LD (IX + d),n

Lade die indiziert adressierte Speicherzelle (IX + d) mit den unmittelbaren Daten n.

Der Inhalt der Speicherzelle, die unmittelbar auf den Op-Code folgt, wird in die Speicherzelle geladen, die durch das Register IX plus dem gegebenen Offset adressiert wird.

Beispiel: LD (IX +4),FF

### LD (IY + d),n

Lade die indiziert adressierte Speicherzelle (IY + d) mit den unmittelbaren Daten n.

Der Inhalt der Speicherzelle, die unmittelbar auf den Op-Code folgt, wird in die Speicherzelle geladen, die durch das Register IY plus dem gegebenen Offset adressiert wird.

Beispiel: LD (IY + 3),BA

### LD (IX + d),r

Lade die indiziert adressierte Speicherzelle (IX + d) aus dem Register r.

Der Inhalt des angegebenen Registers wird in die Speicherzelle geladen, die durch das Register IX plus dem gegebenen Offset adressiert wird.

Beispiel: LD (IX + 1),C

LD (IY + d), r

Lade die indiziert adressierte Speicherzelle (IY + d) aus dem Register r.

Der Inhalt des angegebenen Registers wird in die Speicherzelle geladen, die durch das Register IY plus dem gegebenen Offset adressiert wird.

Beispiel: LD (IY + 3),A

LD A,(nn)

Lade den Akkumulator aus der Speicherstelle nn.

Der Inhalt der Speicherzelle, die durch den Inhalt der beiden Bytes nach dem Op-Code adressiert wird, wird in den Akkumulator geladen. Das untere Byte der Adresse folgt unmittelbar auf den Op-Code.

Beispiel: LD A,(3301)

LD (nn),A

Lade die direkt adressierte Speicherstelle (nn) aus dem Akkumulator.

Der Inhalt der Speicherzelle, die durch den Inhalt der beiden Bytes nach dem Op-Code adressiert wird, wird aus dem Akkumulator geladen. Das untere Byte der Adresse folgt unmittelbar auf den Op-Code.

Beispiel: LD (0321),A

LD (nn),dd

Lade die durch nn adressierten Speicherstellen aus dem Registerpaar dd.

Die untere Hälfte des angegebenen Registerpaares wird in die Speicherzelle geladen, die durch die Speicherstellen adressiert ist, die unmittelbar auf den Op-Code folgen. Die obere Hälfte wird aus der Speicherzelle dahinter geladen. Die untere Hälfte der Adresse nn erscheint unmittelbar hinter dem Op-Code.

Beispiel: LD (040B),BC

LD (nn),HL

Lade die durch nn adressierten Speicherstellen aus dem Register HL.

Der Inhalt des Registers L wird in die Speicherzelle geladen, die durch die Speicherstellen adressiert ist, die unmittelbar auf den Op-Code folgen. Das Register H wird aus der Speicherzelle dahinter geladen. Die untere Hälfte der Adresse nn erscheint unmittelbar hinter dem Op-Code.

Beispiel: LD (40B9),HL

LD (nn),IX

Lade die durch nn adressierten Speicherstellen aus dem Register IX.

Die untere Hälfte des Register IX wird in die Speicherzelle geladen, die durch die Speicherstellen adressiert ist, die unmittelbar auf den Op-Code folgen. Die obere Hälfte wird aus der Speicherzelle dahinter geladen. Die untere Hälfte der Adresse nn erscheint unmittelbar hinter dem Op-Code.

Beispiel: LD (012B),IX

LD (nn),IY

Lade die durch nn adressierten Speicherstellen aus dem Register IY.

Die untere Hälfte des Registers IY wird in die Speicherzelle geladen, die durch die Speicherstellen adressiert ist, die unmittelbar auf den Op-Code folgen. Die obere Hälfte wird aus der Speicherzelle dahinter geladen. Die untere Hälfte der Adresse nn erscheint unmittelbar hinter dem OP-Code.

Beispiel: LD (BD04),IY

LD A,(BC)

Lade den Akkumulator aus der durch das Registerpaar BC indirekt adressierten Speicherstelle.

Der Inhalt der Speicherzelle, die durch den Inhalt des Registerpaares BC adressiert wird, wird in den Akkumulator geladen.

Beispiel: LD A,(BC)

LD A,(DE)

Lade den Akkumulator aus der durch das Registerpaar DE indirekt adressierten Speicherstelle.

Der Inhalt der Speicherzelle, die durch den Inhalt des Registerpaares DE adressiert wird, wird in den Akkumulator geladen.

Beispiel: LD A,(DE)

LD A,I

Lade den Akkumulator aus dem Interruptvektor-Register I.

Der Inhalt des Interruptvektor-Registers I wird in den Akkumulator geladen.

Beispiel: LD A,I



LD I,A

Lade das Interrupt-Register I aus dem Akkumulator.

Der Inhalt des Akkumulators wird in das Interruptvektor-Register I geladen.

Beispiel: LD I,A

LD A,R

Lade den Akkumulator aus dem Memory-Refresh-Register R.

Der Inhalt des Memory-Refresh-Registers wird in den Akumulator geladen.

Beispiel: LD A,R

LD HL,(nn)

Lade das Register HL aus der Speicherzelle nn.

Der Inhalt der Speicherzelle, die durch die Speicherstellen adressiert ist, die unmittelbar auf den Op-Code folgen, wird ins Register L geladen. Das Register H wird aus der Speicherzelle dahinter geladen. Die untere Hälfte der Adresse nn erscheint unmittelbar hinter dem Op-Code.

Beispiel: LD HL,(ØØ24)

LD IX,nn

Lade das Register IX mit den unmittelbaren Daten nn.

Der Inhalt der Speicherzellen, die unmittelbar auf den Op-Code folgen, wird in das Register IX geladen. Das untere Byte erscheint unmittelbar hinter dem Op-Code.

Beispiel: LD IX,BØB1

LD IX,(nn)

Lade das Register IX aus der Speicherzelle nn.

Der Inhalt der Speicherzelle, die durch die Speicherstellen adressiert ist, die unmittelbar auf den Op-Code folgen, wird in die untere Hälfte des Registers IX geladen. Die obere Hälfte wird aus der Speicherzelle dahinter geladen. Die untere Hälfte der Adresse nn erscheint unmittelbar hinter dem Op-Code.

Beispiel: LD IX,(Ø1ØB)

LD IY,nn

Lade das Register IY mit den unmittelbaren Daten nn.

Der Inhalt der Speicherzellen, die unmittelbar auf den Op-Code folgen, wird in das Register IY geladen. Das untere Byte erscheint unmittelbar hinter dem Op-Code.

Beispiel: LD IY,21

**LD IY,(nn)**

**Lade das Register IY aus der Speicherzelle nn.**

**Der Inhalt der Speicherzelle, die durch die Speicherstellen adressiert ist, die unmittelbar auf den Op-Code folgen, wird in die untere Hälfte des Registers IY geladen. Die obere Hälfte wird aus der Speicherzelle dahinter geladen. Die untere Hälfte der Adresse nn erscheint unmittelbar hinter dem Op-Code.**

**Beispiel: LD IY,(500D)**

**Das war's. Bis zur Ausgabe Nr. 8 viel Spass...**

---

Verkaufe Spectrum 48k Komplett-System eingebaut in PC-Gehaeuse, mit abgesetzter PC Tastatur, Beta Disk 4.12 (m. Vision), 3,5" Laufwerk DSDD, 100 Disketten, Originalsoftware, neuer Grundig Datarecorder, Super Rank Xerox Netzteil, Kempston E Drucker-interface, Busverlaengerung mit 2 fach Verteiler, Umschalter fuer Fernseher/Videoausgang, Phillips Gruenmonitor, Doppelport-joystickinterface (Kempston+Sinclair I) fuer 800.- DM bei Bernhard Lutz, Tel. 07272/6868 (bei Wuenschel).

Weiterhin noch ein Ram Turbo Joystick Interface (Sinclair I+II, Cursor, Kempston, ROM-Karten Anschlussmoeglichkeit, durchgef. Bus) nur 30.-. Softwareprogrammierbare DK'troniks Joyst.If., durchgef. Bus: 25.- DM. ZX LPrint III nur 25.- DM.

---

## Suche

**Statistikprogramm "Histo Graph" ev. mit Beschreibung oder ein ähnliches Statistikprogramm für Spektrum 48 kb.**

**Wer kann mir sagen wieso das Musikprogramm "Amadeus 128" beim Spektrum 128 sich beim saven von Noten (auch bei CAT) auf Microdrive immer aufhängt? Mache ich da etwas falsch? Habe das DK'tronics Interface.**

Alfred Pirozzi
Weg in die Vill, 16
39049 Sterzing / Italien

Gojcevic Georg
Lauterach Badweg 06

Lauterach - 1991 - 01 - 14

ZX Spectrum User - Club

---

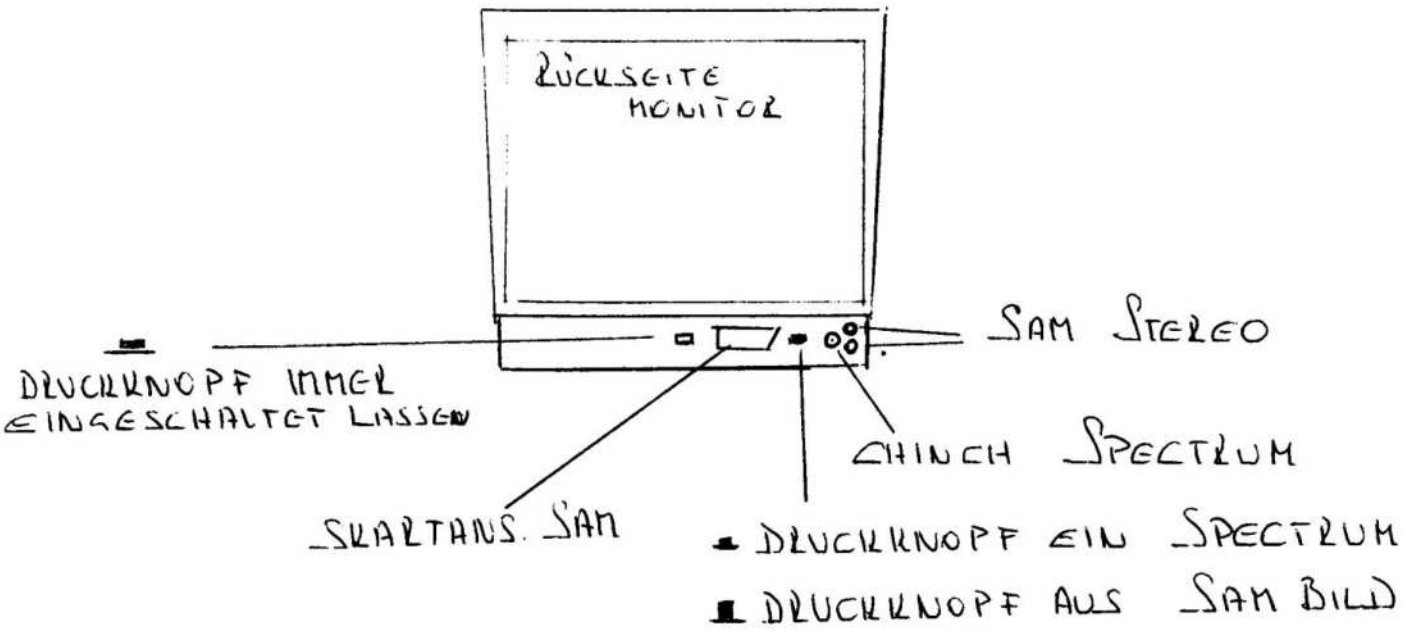
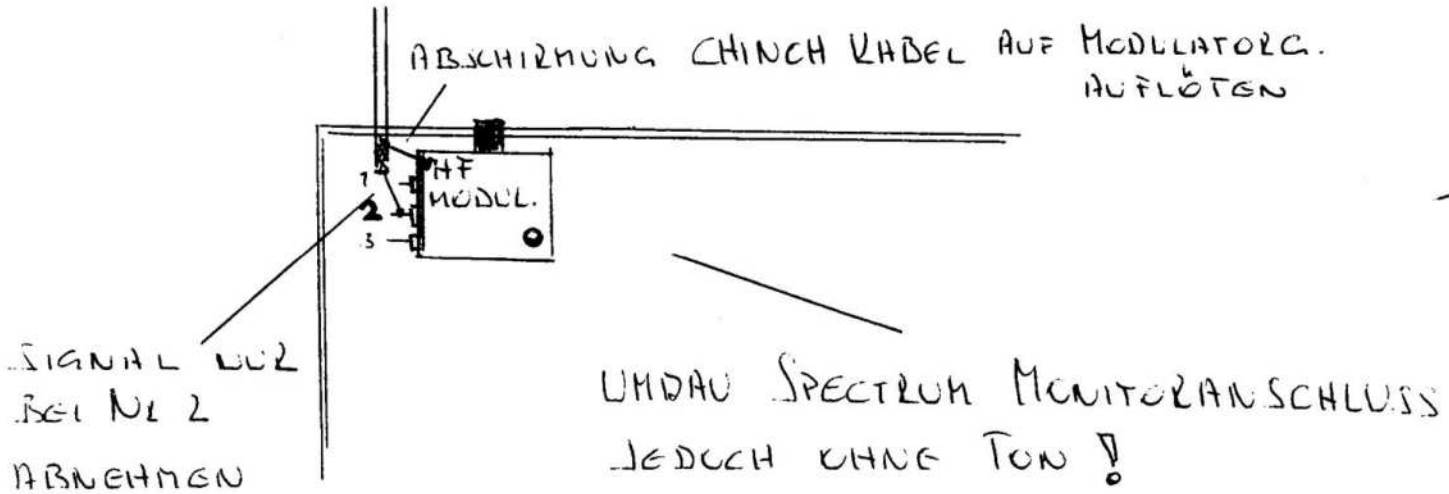
PLATZMANGEL auf dem Computertisch .....????????????????????

Wer für seinen Computer wenig Platz zur Verfügung hat, weiß wovon ich rede. Da ich mir vor einiger Zeit einen SAM zulegte, jedoch auf den Spectrum nicht verzichten kann und will, mußte ich mir bezüglich des Monitoranschlußes etwas einfallen lassen.
2 Monitore und Drucker auf dem Tisch ist einfach zuviel, zumindest bei mir. Also erstand ich für meinen SAM einen PHILIPS 8833-II mit SKARTANSCHLUSS und CHINCH - Buchse.
Den Spectrum 48K schloß ich am CHINCH-Eingang an. Für den SAM steht ja der SKART-Anschluß zur Verfügung.
Wenn ich nun den Spectrum bediene, drücke ich einfach den liksseitigen Knopf ein, beim SAM - Bild drücke ich besagten Knopf wieder und er rastet aus.
Somit kann ich beide Computer an einem Bildschirm bedienen. Es können beide Geräte eingeschaltet sein, ohne daß sie sich stören und ich kann im Bedarfs= fall sofort das jeweilige Gerät benützen.
Da der SAM sogar einen Ausschalter besitzt, ist man als Spectrum FAN nicht unbedingt gewohnt, ist das Abschalten kein Problem.
Um die Leser nicht unnötig zu verwirren, lege ich eine Skitze meines Vor = gangs bei.

SPECTRUM MONITOR ANSCHLUSS

Hier gibt es einen einfachen Trick, wie ich Ihn laut Skitze angewendet habe. Ich schickte den Spectrum + meines Sohnes zwecks Monitoranschluß an die Fa. UFFENKAMP, die mir den Anschluß bewerkstelligte. Da ich von Grund aus neugierig bin, öffnete ich den Spectrum und siehe da, das Ganze hunds-einfach.
Der 48K + ist Issue 6A. Nun mußte mein alter 48K Issue 2 herhalten, was soll ich sagen :FUNKTIONIERT Prima das Ganze und die Farbqualität ist gar nicht schlecht.
Nun noch viel Vergnügen beim Löten .....

Ps: Wer einen Trick mit dem Ton hat, ( Spectrum - Monitor ) bitte melden!
Das Ganze funktioniert auch monochrom zumindest bei mir!

# SKIZZE MONITORANSCHUSS SAM / SPECTRUM

PHILIPS 8833 - II